TR/RP No. 69

UNITED STATES
NAVAL POSTGRADUATE SCHOOL

AN ITERATIVE SOLUTION
TO THE
GENERALIZED EIGENVALUE - EIGENVECTOR PROBLEM

by

R. D. Brunell

July 1966

TECHNICAL REPORT/RESEARCH PAPER NO. 69

Distribution of this document is unlimited

AN ITERATIVE SOLUTION
TO THE
GENERALIZED EIGENVALUE - EIGENVECTOR PROBLEM

by

R. D. Brunell

July 1966

TECHNICAL REPORT/RESEARCH PAPER NO. 69

UNITED STATES NAVAL POSTGRADUATE SCHOOL

Monterey, California

Rear Admiral E. J. O'Donnell, USN, Superintendent

R. F. Rinehart, Academic Dean

ABSTRACT:

This paper presents an approach to the solution of what is usually called "the generalized eigenvalue problem." The basic format of the method is similar to that presented by I. Tarnove (3), but with revisions in the root-finding and scaling procedures. The capability of calculating the associated eigenvectors has also been added to the previously-published algorithm.

This method uses an iterative root-finding technique and will converge with almost any first estimate. To preserve accuracy, the original matrix is used throughout for the evaluation of the function and scaling is done by powers of two.

A FORTRAN-63 program for the CDC-1604, using this method, has been written for matrices having polynomial elements. The code was evaluated using a number of trial problems, including pathological matrices, and some of these results have been incorporated into this report.

Prepared by: R. D. Brunell

Mathematician, Computer Facility

Approved by:

D. G. Williams

Head, Computer Facility

Released by:

C. E. Menneken

Dean of

Research Administration

U. S. Naval Postgraduate School Technical Report/Research Paper No. 69

July 1966

UNCLASSIFIED

[illegible]

[illegible]

[illegible] [illegible]

[illegible] [illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible] [illegible]

[illegible] [illegible]

[illegible] [illegible]

[illegible]

[illegible]

[illegible]

TABLE OF CONTENTS

F4-NPGS-GENEIG, A Method for the Solution of the "Generalized Eigenvalue Problem" with Polynomial Elements, June 1966.

AN ITERATIVE SOLUTION TO THE GENERALIZED EIGENVALUE - EIGENVECTOR PROBLEM

I. Statement of the Problem

The "generalized eigenvalue problem" may be stated as follows:

Find those values of the complex parameter, z, which will provide a non-trivial solution to the system of linear equations,

$$
\begin{aligned}
A_{11}(z)X_1 + A_{12}(z)X_2 + \dots + A_{1n}(z)X_n &= 0 \\
A_{21}(z)X_1 + A_{22}(z)X_2 + \dots + A_{2n}(z)X_n &= 0 \\
\vdots \qquad\quad \vdots \qquad\qquad\quad &\vdots \\
A_{n1}(z)X_1 + A_{n2}(z)X_2 + \dots + A_{nn}(z)X_n &= 0.
\end{aligned}
\tag{1}
$$

This can be written more concisely as

$$A(z)X = 0 \tag{2}$$

where A is an n x n matrix, the elements of which are arbitrary complex functions of z, and X is a complex column vector of n elements. The set $\{z_i : A(z_i)X_i = 0\}$ is the set of eigenvalues and the X_i are called the associated eigenvectors.

The special case where the $A_{ij}(z)$'s are polynomials in z of degree 2 or less is encountered most often in practical work. The rest of the discussion in this paper will be confined to problems of this type. However, it should be pointed out that the general method described here is not so restricted.

Under the above restriction we can rewrite A(z) as

(3) $$A(z) = A_o z^2 + A_1 z + A_2$$

where the A_i's are constant complex matrices. We will need this form in the later discussion on scaling.

II. Method of Solution

The heart of the method described here is the iterative root-finding technique. After a literature search, in which much background information was gained from Wilkinson (1), and some experimentation, it was decided that a method developed by J. F. Traub (2) would be used in this code. Muller's method (7) has been mentioned by a number of authors in connection with iterative solutions in the complex plane; however, Traub's technique was chosen because of its simplicity and ease of computation. The methods are similar; both are of second order and have the same asymptotic error constant. The main difference is that Traub uses the Newtonian form of the interpolating polynomial and Muller uses the Lagrangian form.

In essence the technique is to perform a quadratic fit using the last three iterants and their functional values. This algorithm is defined as follows:

(4) $$z_{i+1} = z_i - 2f_i / (w \pm \{w^2 - 4f_i \; f[\, z_i, z_{i-1}, z_{i-2}]\}^{1/2})$$

where

(5) $$w = f[\, z_i, z_{i-1}] + (z_i - z_{i-1}) f[\, z_i, \, z_{i-1}, z_{i-2}]$$

and

(6) $$f[z_i, z_{i-1}, z_{i-2}] \equiv (f[z_i, z_{i-1}] - f[z_{i-1}, z_{i-2}])/(z_i - z_{i-2})$$

The variable sign in equation (4) is chosen so that the denominator has the larger magnitude.

? $f(z) = \det |A_0 z^2 + A_1 z + A_2|$?

Convergence is determined by either of the two tests,

$$|z_i - z_{i-1}| / |z_i| < e_1 \tag{7}$$

or

$$|f_i| < e_2 \quad . \tag{8}$$

If either of these inequalities is met the process is considered to have converged to a root.

In order to avoid convergence to a previously-calculated root, a method of suppressing zeros, first suggested by G. E. Forsythe (Reference 3, page 164) was incorporated into the code. The first root, z_1, is found as a zero of f(z) and the remaining roots, z_i, are found as zeros of $g_i(z)$, where

$$g_i(z) \equiv f(z) / \prod_{k=1}^{i} (z - z_k) \quad , \qquad i > 1 \quad . \tag{9}$$

One of the problems encountered in a program of this type is floating-point exponential overflow when working with matrices of large order. To alleviate this problem an automatic scaling feature was devised. This scaling is done in two parts, _a priori_ and _a posteriori_. The _a priori_ scaling is done at the beginning of the problem and is carried out in the following manner. The coefficients of each polynomial element in a row of the matrix A are summed and a maximum is found. In a binary computer this maximum may be expressed exactly as $bx2^j$. Then each coefficient in every polynomial in that row is divided by 2^j. This operation is carried out for each row in the matrix.

A posteriori scaling is done only when an overflow occurs. If, at the time of the overflow, the estimate to the root being calculated is some number, say $cx2^k$, then A(z) (defined in equation (3)) is replaced by

$$A'(z') = A_o z^2 + \frac{A_1}{2^k} z + \frac{A_2}{(2^k)^2} \quad , \tag{10}$$

and a root, z', is then found for the new problem. The root of the original problem is then calculated as

$$z_i = 2^k z' \tag{11}$$

The scaled problem is started by taking the last three iterants in the original problem and dividing them by 2^k. After a root is found the matrix is returned to its original form and the program proceeds to find the next root.

In many cases it is desirable to obtain the eigenvectors associated with the eigenvalues. An option to perform these calculations is available in the program. Because there exists no procedure which will absolutely assure the finding of an eigenvector, the code will make 2n attempts, where n is the order of the matrix, at computing an eigenvector for each eigenvalue. This procedure is carried out as outlined below.

An assumption is made that the elements of the vector may be written as multiples of a single element in the vector. One possible representation of this is

$$X_i^T = (x_{1i},\ b_1 x_{2i}, b_2 x_{3i}, \ldots,\ b_{n-1} X_{ni}) \quad . \tag{12}$$

Using this assumption the code makes its first attempt by setting $X_{1j}=1$ and solving the system of equations (13):

$$\begin{pmatrix} a_{22} & a_{23} & \cdots & a_{2n} \\ a_{32} & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & & \vdots \\ a_{n2} & a_{n3} & \cdots & a_{nn} \end{pmatrix} \begin{pmatrix} X_{2i} \\ X_{3i} \\ \vdots \\ X_{ni} \end{pmatrix} = \begin{pmatrix} -a_{21} \\ -a_{31} \\ \vdots \\ -a_{n1} \end{pmatrix} , \tag{13}$$

where a_{ij} is the function A_{ij} evaluated at z_i.

The expression, A(z) X, is then calculated using this vector and the result compared to the zero vector. If the vector, X, is found to be unacceptable a second attempt is made by replacing a_{2k} by a_{1k}, $k=1,2,\ldots,n$, in (13). If this fails then x_{2i} is set equal to unity and the system (14) is solved,

$$(14) \qquad \begin{pmatrix} a_{11} & a_{13} & \cdots & a_{1n} \\ a_{31} & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n3} & \cdots & a_{nn} \end{pmatrix} \begin{pmatrix} X_{1i} \\ X_{3i} \\ \vdots \\ X_{ni} \end{pmatrix} = \begin{pmatrix} -a_{12} \\ -a_{32} \\ \vdots \\ -a_{n2} \end{pmatrix}$$

and checked as indicated above. A fourth attempt may be made by replacing, in (14), a_{3k} by a_{2k} for $k=1,2,\ldots,n$. The process continues for all X_{ji} or until an eigenvector is found. If no eigenvector is found a statement to that effect is printed along with the results of the final attempt.

III. General Discussion of Method

A code, using the method described in this paper, has been written in FORTRAN-63 for use on a CDC 1604. This program has been checked out on a variety of problems, several of which are shown in the Section IV. The accuracy of the process has been shown to be quite good. This may be attributed to several factors. First, only the original input matrix is used for the evaluation of the function each time, and, second, all scaling is done by powers of two (Reference (8), page 98). The method has been shown to be reasonably fast, although a good relative comparison is impossible because the number of iterations required is a function of

the estimate provided. Unfortunately this information generally has not been included in previous literature on this subject. Because the number of computations necessary to determine the next estimate to the root is so small compared to those required to evaluate the determinant, the time required for each iteration can be considered substantially equal to the time necessary for the latter. The speed of the program is further enhanced by the fact that through the use of good estimates only eigenvalues of interest need be computed, although convergence seems to result from almost arbitrary estimates. Also because of the feature which forces avoidance of previously found zeros, it is possible to restart the program and by-pass any such prior calculations.

IV. Numerical Examples

Most of the examples used for this section have been taken from papers on the standard eigenvalue problem. This was necessary because of the lack of comparative data on the generalized problem. However, these examples should be adequate to demonstrate the ability of this method to find pathological roots and also to get some comparison as to its accuracy. All problems, using the method described in this report, were run on a CDC 1604 which has a 48-bit word length.

[illegible]

Example 1

(See Parlett (5)). These results were obtained using Laguerre's method on an IBM 7090 which has a 36-bit word. A is defined as

$$\begin{pmatrix}
611-z & 196 & -192 & 407 & -8 & -52 & -49 & 29 \\
196 & 899-z & 113 & -192 & -71 & -43 & -8 & -44 \\
-192 & 113 & 899-z & 196 & 61 & 49 & 8 & 52 \\
407 & -192 & 196 & 611-z & 8 & 44 & 59 & -23 \\
-8 & -71 & 61 & 8 & 411-z & -599 & 208 & 208 \\
-52 & -43 & 49 & 44 & -599 & 411-z & 208 & 208 \\
-49 & -8 & 8 & 59 & 208 & 208 & 99-z & -911 \\
29 & -44 & 52 & -23 & 208 & 208 & -911 & 99-z
\end{pmatrix}.$$

Table of Eigenvalues

Actual (8 digits)	Parlett (5)	Present Method
1020.0490	1020.0500	1020.0490
1020.0000	1019.9997	1020.0000
1019.9020	1019.9019	1019.9019
1000.0000	1000.0001	1000.0000
1000.0000	999.99999	1000.0000
.098048640	.098045509	0.098048646
0.0	-0.0000094	-0.000000003
-1020.0490	-1020.0490	-1020.0490

Total running time: 16 seconds
Average number of iterations per eigenvalue: 10.5

[illegible]

[illegible]

[illegible]

[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]

[illegible]

[illegible]

Example 2

(See Barlow and Jones (4)). Their work was done on an IBM 7040 (36-bit word) using an extension of the classical secant method to the complex field. This paper (4) also gives results obtained by Eberlein (6) using a Jacobi-like method.

$$A = \begin{pmatrix} 15-z & 11 & 6 & -9 & -15 \\ 1 & 3-z & 9 & -3 & -8 \\ 7 & 6 & 6-z & -3 & -11 \\ 7 & 7 & 5 & -3-z & -11 \\ 17 & 12 & 5 & -10 & -16-z \end{pmatrix}$$

Table of Eigenvalues

Actual (6)	Eberlein (6)	Barlow & Jones (4)	Present Method
-1.0	-.999	-1.0000004	-.999999999
1.50016 + 3.57064i	1.505 + 3.57i	1.49850 + 3.57027i	1.50001 + 3.57073i
1.50016 + 3.57064i	1.495 + 3.57i	1.50150 + 3.57110i	1.50003 + 3.57069i
1.50016 - 3.57064i	1.505 - 3.57i	1.49996 - 3.57127i	1.50001 - 3.57074i
1.50016 - 3.57064i	1.495 - 3.57i	1.49864 - 3.57023i	1.49999 - 3.57070i

Total running time: 4.38 seconds
Average number of iterations per eigenvalue: 6

<u>Example 3</u>

(See Parlett (5)). This example was used to demonstrate the ability of the code to extract only a few desired roots.

$$A = (B - zI)$$

$$B = \begin{pmatrix} C & 2C \\ 4C & 3C \end{pmatrix} \qquad C = \begin{pmatrix} 5D & -D \\ 5D & D \end{pmatrix} \qquad D = \begin{pmatrix} -2 & 2 & 2 & 2 \\ -3 & 3 & 2 & 2 \\ -2 & 0 & 4 & 2 \\ -1 & 0 & 0 & 5 \end{pmatrix}$$

The true eigenvalues of this system (Eberlein (6)) are $60 \pm 20i$, $45 \pm 15i$, $30 \pm 10i$, $15 \pm 5i$, $-12 \pm 4i$, $-9 \pm 3i$, $-6 \pm 2i$, $-3 \pm i$.

Initial Guess	Eigenvalue Found
-2.5 - 0.625i	-3.0000006 - 9.9999947i
-2.5 + 0.625i	-3.0000005 + 1.0000000i
-5.0 - 1.25i	-6.0000055 - 1.9999948i
-5.0 + 1.25i	-5.9999975 + 2.0000003i

Total running time: 55.28 seconds
Average number of iterations per eigenvalue: 17.5

Example 4

In this example both the eigenvalues and eigenvectors were found.

$$A = \begin{pmatrix} 1-z & -2 & 3 & -2 \\ 1 & 5-z & -1 & -1 \\ 2 & 3 & 2-z & -2 \\ 2 & -2 & 6 & -3-z \end{pmatrix}$$

Table of Eigenvalues and Eigenvectors

True Values		Computed Values	
Eigenvalues	Eigenvectors	Eigenvalues	Eigenvectors
-1	1 0 0 1	-1.00000000	1.0 0.0 0.0 1.0
2	1 -6 -9 -8	1.999939	1.0 -5.9993 -8.9991 -7.9993
2	1 -6 -9 -8	2.00007	1.0 -6.0007 -9.001 -8.0008
2	1 -6 -9 -8	1.99998	1.0 -5.9998 -8.9998 -7.9998

Total running time: 3.83 seconds
Average number of iterations per eigenvalue: 8.25

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]

[illegible]

Example 5

This is an example of the generalized problem.

$$A = \begin{pmatrix} 2z^2+14z-88 & -z^2-13z-22 & z^2+2z-99 \\ -3z^2+21z-36 & 6z^2-6z-36 & 4z^2-48z+108 \\ 8z^2+8z-160 & z^2+7z+10 & -z^2+4z+45 \end{pmatrix}$$

Table of Eigenvalues and Eigenvectors

True Values		Computed Values	
Eigenvalues	Eigenvectors	Eigenvalues	Eigenvectors
-11	1 -29/6 153/40	-10.99999999	1.0 -4.833333333 3.82500000
-5	10/33 1 -27/154	-5.0	0.3030303030 1.0 -1.753246753
-2	0 1 0	-2.0	0.0 1.0 0.0
3	0 1 -5/6	3.0	-0.0000000001 1.0 -0.8333333333
4	1 0 0	4.0	1.0 0.0 0.0
9	0 0 1	9.0	0.0 0.0 1.0

Total running time: 3.85 seconds
Average number of iterations per eigenvalue: 4.0

Example 6

This final example of the generalized problem is constructed so that it can also be solved as a standard problem. It is defined as follows,

i.) $A \equiv (B+zC) = 0,$

where B and C are 12x12 matrices with constant elements. By multiplying on the left by C^{-1} a transformation is made to the standard problem,

ii.) $A' \equiv (BC^{-1}+zI) = 0.$

This problem has been solved in both forms by the technique discussed here. In addition, it was solved using the algorithm in (5), so that the results might be compared.

The matrices B and C are defined on the following page.

The matrices B and C were defined as:

$$\begin{pmatrix} 10 & -15 & 1 & -2 & -81 & 91 & -69 & 14 & 62 & -36 & -20 & 99 \\ 22 & -46 & -25 & -85 & -3 & 89 & 27 & 53 & -93 & -34 & 52 & 19 \\ 24 & 48 & -22 & -97 & -76 & -64 & -15 & 24 & 49 & -32 & 30 & -19 \\ -42 & -93 & -6 & 61 & 7 & 16 & 39 & -53 & -71 & 57 & 0 & -74 \\ 37 & -39 & -81 & 16 & -6 & 91 & 60 & -81 & 49 & 60 & 14 & -6 \\ -77 & -6 & 11 & -42 & 27 & 53 & 18 & -70 & -90 & -15 & 21 & -81 \\ 99 & -72 & 56 & 69 & 98 & 31 & 71 & 18 & -44 & 48 & -63 & -21 \\ -96 & -91 & -5 & 7 & 18 & 20 & -94 & -56 & 69 & -60 & -18 & -84 \\ -89 & 14 & -63 & -10 & -17 & -18 & 57 & 84 & -25 & 12 & 58 & -44 \\ -85 & -36 & 53 & 53 & 53 & -59 & -38 & 62 & 8 & -17 & -16 & 11 \\ 28 & 69 & -88 & 33 & -70 & 79 & -56 & -5 & 90 & -31 & -1 & 85 \\ -63 & -40 & -48 & -3 & 49 & -69 & -18 & -72 & 52 & -20 & 12 & -90 \end{pmatrix} = B$$

$$\begin{pmatrix} -81 & -72 & -4 & -96 & 24 & -82 & 66 & 14 & -76 & 14 & 13 & 87 \\ 29 & -20 & 68 & 26 & -46 & -20 & 89 & 81 & -86 & -12 & -92 & 57 \\ 0 & 57 & 39 & 66 & -84 & -40 & 32 & 61 & -98 & -96 & 64 & 64 \\ 5 & -4 & -25 & 26 & -44 & 44 & -37 & 63 & 45 & 66 & 75 & 66 \\ -91 & 26 & -64 & -94 & 26 & 25 & 39 & -22 & 71 & 64 & 91 & 42 \\ 0 & -4 & -87 & -77 & 42 & 35 & -74 & -99 & -81 & -42 & 43 & -76 \\ 0 & 69 & -62 & 56 & 86 & 88 & 76 & 36 & -84 & -93 & 76 & -65 \\ -69 & -65 & 95 & -55 & 18 & 27 & -26 & -8 & -40 & 59 & -29 & 80 \\ 25 & 57 & 29 & 88 & -67 & 48 & 18 & -82 & 65 & 69 & -33 & 54 \\ -9 & -83 & -73 & 12 & -30 & -18 & 28 & -35 & 5 & 41 & 34 & 37 \\ -91 & -42 & 27 & 30 & 4 & -86 & 29 & 99 & 55 & -84 & 29 & 9 \\ -17 & -56 & -90 & -49 & 20 & -59 & -6 & 20 & -18 & 2 & 73 & -83 \end{pmatrix} = C$$

These three cases were run with a relative error bound of 10^{-6} and the results from all of them were identical to six or seven significant figures. The eigenvalues found are listed in the table below.

Table of Eigenvalues
.956212
-2.67567
- .0213507
- .466901
- .626829 $\pm$ 2.56009
- .274574 $\pm$ .327271
.254714 $\pm$ .457145
- .796302 $\pm$ 1.04215

Total time for present method: 104 seconds
Average number of iterations per eigenvalue: 17.2

[illegible]

[illegible]	[illegible]

[illegible]

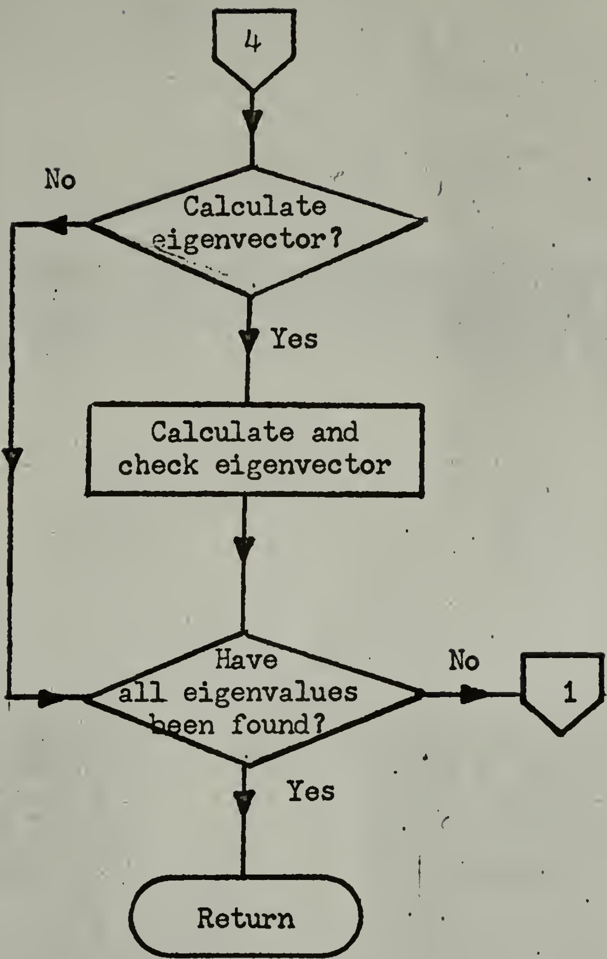
V. Areas of Further Investigation

Some additional features are being considered for inclusion in the present code. However, at the time of this report all problems involving these revisions had not been resolved.

It is felt that some process which could quickly determine the approximate location of eigenvalues may be useful, especially when large order problems are undertaken. It has not, as yet, been determined what method might be most profitable for this purpose.

An automatic entry into double precision would be desirable for parts of the computations. But it will be necessary to devise some general test to indicate when this move could best be made without wasting computer time.

A test to find when the domain of indeterminancy has been reached in the search for a root would be a valuable addition to those existing tests for acceptance of a root.


4
No
Calculate eigenvector?
Yes
Calculate and check eigenvector
Have all eigenvalues been found?
No
1
Yes
Return

APPENDIX II

```
      SUBROUTINE GENEIG
C
C         THIS IS THE CONTROL SUBROUTINE
C         THE FOLLOWING INFORMATION MUST BE SETUP IN COMMON FOR GENEIG
C
C         INPUT INFORMATION
C         THE ARRAY A MUST CONTAIN THE COEFFICIENT MATRICES
C         THE ARRAY EIGG MUST CONTAIN THE FIRST GUESSES
C         NN=RANK OF MATRICES
C         IORD=ORDER OF POLYNOMIAL ELEMENTS
C         IE=NUMBER OF FIRST EIGENVALUE TO BE FOUND
C         NEIG=NUMBER OF LAST EIGENVALUE TO BE FOUND
C         IMAX=MAXIMUM NUMBER OF ITERATIONS TO BE ALLOWED
C         ISAC=FLAG FOR APRIORI SCALING, ISAC=0 SCALING WILL OCCUR
C         IPRINT=FLAG FOR PRINT AT EVERY ITERATION,IPRINT=0 NO PRINTING
C         IVEC=FLAG FOR COMPUTING EIGENVECTORS,IVEC=0 NO EIGENVECTORS
C         EP1=RELATIVE ERROR BOUND ON EIGENVALUE
C         EP2=ABSOLUTE ERROR BOUND ON FUNCTION
C         EP3=RELATIVE ERROR BOUND ON EIGENVECTOR
C
C         OUTPUT INFORMATION
C         THE ARRAY EVAL WILL CONTAIN THE COMPUTED EIGENVALUES
C         THE ARRAY EVEC WILL CONTAIN THE ASSOCIATED EIGENVECTORS
C
      DIMENSION AE(20,20)
      COMMON A(20,20,3),EIGG(40,2),EVAL(40),EVEC(20,40),NN,IORD,IE,NEIG,
     1IMAX,ISAC,IPRINT,IVEC,EP1,EP2,EP3
      COMMON /GBLOC2/AE,XI,XI1,XI2,FI,FI1,FI2,FX0,HI2,YXI1,YXI2,XS2,II
     1,IJ,IKJ  ,CC1,CC2,CC3,CC4,CC5,DD1,DD2,DD3,DD4,DD5,DD6,DD7,DD8
      TYPE COMPLEX AE,XI,XI1,XI2,FI,FI1,FI2,HI2 ,CC1,CC2,CC3,CC4,CC5
      TYPE COMPLEX A,EIGG,EVAL,EVEC
      XS2=1.0
      DD8=0.0
      WRITE (51,30)
   30 FORMAT(1H1)
      IF(ISAC) 20,10,20
C
C         APRIORI SCALING IF DESIRED
C
   10 CALL SCALE1
   20 DO 1000 IJ=IE,NEIG
      DD1=0.0
      II=1
```

```
C
C        CALCULATE GUESSES TO EIGENVALUES
C
      XI =EIGG(IJ,1)+EIGG(IJ,1)*EIGG(IJ,2)
      XI1=EIGG(IJ,1)
      XI2=EIGG(IJ,1)-EIGG(IJ,1)*EIGG(IJ,2)
C
C        CALCULATE NEXT GUESS AT EIGENVALUE
C
   60 CALL CALNEWX
C
C         EVALUATE FUNCTION AT NEW ESTIMATE TO EIGENVALUE
C
   40 CALL FUNCEV(XI,FI)
      IF(DD8) 50,50,200
  200 DD8=0.0
C
C         APOSTERIORI SCALING IF NECESSARY
C
      CALL SCALE2
      GO TO 60
   50 IF(IPRINT) 70,90,70
C
C         PRINT EACH ITERATION
C
   70 WRITE (51,80),II,XI,CC5
   80 FORMAT(2X,I5,10H ITERATION ,3H X=,C(E22.11,E22.11),7H  F(X)=,
     1C(E22.11,E22.11))
C
C         DETERMINE IF NEW ESTIMATE TO EIGENVALUE IS ACCEPTABLE
C
   90 CALL ACCEPT
      II=II+1
C
C         DETERMINE IF PROBLEM HAS CONVERGED
C
      GO TO (100,130),IKJ
  100 IF(II-IMAX)60,110,110
  110 WRITE (51,120)
  120 FORMAT(4X,20HMAXIMUM ITERATIONS    )
  130 WRITE (51,140) ,IJ,XI,CC5
  140 FORMAT(//4X,14HEIGENVALUE NO.,I3,2H =,C(E22.11,E22.11),2X,5HF(X)=,
     1C(E22.11,E22.11))
```

```
C
C     CALCULATE GUESSES TO EIGENVALUES
C
      XI =EIGG(IJ,1)+EIGG(IJ,1)*EIGG(IJ,2)
      XI1=EIGG(IJ,1)
      XI2=EIGG(IJ,1)-EIGG(IJ,1)*EIGG(IJ,2)
C
C     CALCULATE NEXT GUESS AT EIGENVALUE
C
   60 CALL CALNEWX
C
C     EVALUATE FUNCTION AT NEW ESTIMATE TO EIGENVALUE
C
   40 CALL FUNCEV(XI,FI)
      IF(DD8) 50,50,200
  200 DD8=0.0
C
C     APOSTERIORI SCALING IF NECESSARY
C
      CALL SCALES
      GO TO 60
   50 IF(IPRINT) 70,90,70
C
C     PRINT EACH ITERATION
C
   70 WRITE (51,80),II,XI,CC5
   80 FORMAT(2X,I5,10H ITERATION ,3H X=,C(E22.11,E22.11),7H  F(X)=,
     1C(E22.11,E22.11))
C
C     DETERMINE IF NEW ESTIMATE TO EIGENVALUE IS ACCEPTABLE
C
   90 CALL ACCEPT
      II=II+1
C
C     DETERMINE IF PROBLEM HAS CONVERGED
C
      GO TO (100,130),IKJ
  100 IF(II-IMAX)60,110,110
  110 WRITE (51,120)
  120 FORMAT(4X,20HMAXIMUM ITERATIONS    )
  130 WRITE (51,140) ,IJ,XI,CC5
  140 FORMAT(//4X,14HEIGENVALUE NO.,I3,2H =,C(E22.11,E22.11),2X,5HF(X)=,
     1C(E22.11,E22.11))
```

```
  900 EVAL(IJ)=XI*XS2
      IF(DD1) 930,940,930
C
C          INVERSE TRANSFORMATION IF APOSTERIORI SCALING HAD OCCURED
C
  930 CALL ROMAT
  940 IF (IVEC)  950,1000,950
C
C          COMPUTE EIGENVECTORS IF DESIRED
C
  950 CALL EVEK
 1000 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE CALNEWX
C
C         THIS SUBROUTINE DETERMINES THE VALUE OF THE NEXT ITERANT
C         USING TRAUBS METHOD
C
      DIMENSION AE(20,20)
      COMMON A(20,20,3),EIGG(40,2),EVAL(40),EVEC(20,40),NN,IORD,IE,NEIG,
     1IMAX,ISAC,IPRINT,IVEC,EP1,EP2,EP3
      COMMON /GBLOC2/AE,XI,XI1,XI2,FI,FI1,FI2,FX0,HI2,YXI1,YXI2,XS2,II
     1,IJ,IKJ  ,CC1,CC2,CC3,CC4,CC5,DD1,DD2,DD3,DD4,DD5,DD6,DD7,DD8
      TYPE COMPLEX A,EIGG,EVAL,EVEC
      TYPE COMPLEX AE,XI,XI1,XI2,FI,FI1,FI2,HI2 ,CC1,CC2,CC3,CC4,CC5
      TYPE COMPLEX HI1,HI12,DELX1,DELX2,W,T,XC,X,XX1,XX2
      IF(II-1) 10,10,50
C
C         FIND INITIAL VALUES
C
   10 CALL FUNCEV(XI2,FI2)
      IF (IPRINT) 35,15,35
   35 WRITE (51,210) ,XI2,CC5
  210 FORMAT(//6X,12H1ST GUESS    ,2HX=,C(E22.11,E22.11),7H  F(X)=,
     1C(E22.11,E22.11))
   15 CALL FUNCEV(XI1,FI1)
      IF (IPRINT) 25,45,25
   25 WRITE (51,220) ,XI1,CC5
  220 FORMAT(6X,12H2ND GUESS    ,2HX=,C(E22.11,E22.11),7H  F(X)=,
     1C(E22.11,E22.11))
   45 CALL FUNCEV(XI,FI)
      IF (IPRINT) 55,65,55
   55 WRITE (51,230) ,XI,CC5
  230 FORMAT(6X,12H3RD GUESS    ,2HX=,C(E22.11,E22.11),7H  F(X)=,
     1C(E22.11,E22.11))
   65 IF(DD8) 40,40,20
C
C         DETERMINE IF APOSTERIORI SCALING IS NECESSARY
C
   20 CALL SCALE2
      DD8=0.0
   40 HI2=(FI1-FI2)/(XI1-XI2)
C
C         COMPUTE NEXT ITERANT
C
   50 DELX1=XI-XI1
```

```
      DELX2=XI-XI2
      HI1=(FI-FI1)/DELX1
      HI12=(HI1-HI2)/DELX2
      W=HI1+DELX1*HI12
      XC=CSQRT(W*W-4.0*FI*HI12)
      XX1=W+XC
      XX2=W-XC
      XC1=CABS(XX1)
      XC2=CABS(XX2)
C
C        DETERMINE WHICH ROOT IS TO BE USED
C
      IF (XC1-XC2) 90,90,100
   90 T=XI-(2.0*FI)/XX2
      GO TO 110
  100 T=XI-(2.0*FI)/XX1
C
C        SET VALUES FOR NEXT ITERATION
C
  110 XI2=XI1
      XI1=XI
      XI=T
      FI1=FI
      HI2=HI1
  190 RETURN
      END
```

```
      SUBROUTINE CDTERM (N,B,DET,KER,M)
C
C          THIS SUBROUTINE CALCULATES THE DETERMINANT USING THE
C          GAUSS METHOD WITH PIVOTING
C
      DIMENSION B(2,20,20),DET(2)
      C=1.0
      DO 35 L=1,N
      KP=0
      Z=0.0
      DO 12 K=L,N
C
C          FIND MAXIMUM ELEMENT IN COLUMN
C
      BMAG=B(1,K,L)**2+B(2,K,L)**2
      IF(Z-BMAG) 11,12,12
   11 Z=BMAG
      KP=K
   12 CONTINUE
      IF(L-KP)13,30,30
C
C          SIGN CHANGE IF PIVOTING IF NECESSARY
C
   13 C=-C
      DO 14 J=L,N
C
C          INTERCHANGE ROWS
C
      TEMP=B(1,L,J)
      Z=B(2,L,J)
      B(1,L,J)=B(1,KP,J)
      B(2,L,J)=B(2,KP,J)
      B(1,KP,J)=TEMP
      B(2,KP,J)=Z
   14 CONTINUE
   30 IF(L-N) 31,40,40
   31 LP1=L+1
      DO 35 K=LP1,N
C
C          GAUSS METHOD
C
      IF(B(1,K,L)) 33,46,33
   46 IF(B(2,K,L)) 33,35,33
```

```
33 Z=B(1,L,L)**2+B(2,L,L)**2
   IF(Z) 43,44,43
43 TEMP=(B(1,K,L)*B(1,L,L)+B(2,K,L)*B(2,L,L))/Z
   BMAG=(B(1,L,L)*B(2,K,L)-B(1,K,L)*B(2,L,L))/Z
   DO 34 J=LP1,N
   B(1,K,J)=B(1,K,J)-(TEMP*B(1,L,J)-BMAG*B(2,L,J))
   B(2,K,J)=B(2,K,J)-(TEMP*B(2,L,J)+BMAG*B(1,L,J))
34 CONTINUE
35 CONTINUE
40 DET(1)=B(1,1,1)
   DET(2)=B(2,1,1)
   DO 41 K=2,N
   BMAG=(DET(1)*B(1,K,K)-DET(2)*B(2,K,K))
   DET(2)=(DET(1)*B(2,K,K)+DET(2)*B(1,K,K))
   DET(1)=BMAG
41 CONTINUE
   IF (C) 50,44,60
50 DET(1)=-DET(1)
   DET(2)=-DET(2)
60 KER=1
   RETURN
44 KER=2
   RETURN
   END
```

```
      SUBROUTINE JORCOM (A,N,X,KR,KC)
C
C         THIS SUBROUTINE COMPUTES THE EIGENVECTOR USING THE
C         GAUSS-JORDAN METHOD
C
      DIMENSION A(KR,KC),X(KR)
      TYPE COMPLEX A,X,XY
C
      K=N+1
   11 IF(K-1) 13,6,15
   15 D=0.0
C
C         FIND MAXIMUM ELEMENT IN COLUMN
C
      DO 2 I=2,K
      II = I-1
      C1=CABS(A(II,1))
      IF(C1-D) 2,2,3
    3 L=II
      D=C1
    2 CONTINUE
    4 IF(L-1) 5,6,5
C
C         INTERCHANGE ROWS
C
    5 DO 7 J=1,K
      XY=A(L,J)
      A(L,J)=A(1,J)
      A(1,J)=XY
    7 CONTINUE
C
C         SETUP SOLUTION VECTOR
C
    6 DO 8 I=1,N
      X(I)=A(I,1)
    8 CONTINUE
      IF (K-1) 12,13,12
C
C         GAUSS-JORDAN METHOD
C
   12 DO 10 J=2,K
      IJ=J-1
      XY=A(1,J)/X(1)
```

```
      DO 9 I=2,N
      A(I-1,IJ)=A(I,J)-X(I)*XY
    9 CONTINUE
      A(N,IJ)=XY
   10 CONTINUE
      K=K-1
      GO TO 11
   13 RETURN
      END
```

```
      SUBROUTINE CALFUNC(AZ)
C
C         THIS SUBROUTINE EVALUATES THE POLYNOMIAL ELEMENTS AT XI
C
      DIMENSION AE(20,20)
      COMMON A(20,20,3),EIGG(40,2),EVAL(40),EVEC(20,40),NN,IORD,IE,NEIG,
     1IMAX,ISAC,IPRINT,IVEC,EP1,EP2,EP3
      COMMON /GBLOC2/AE,XI,XI1,XI2,FI,FI1,FI2,FX0,HI2,YXI1,YXI2,XS2,II
     1,IJ,IKJ  ,CC1,CC2,CC3,CC4,CC5,DD1,DD2,DD3,DD4,DD5,DD6,DD7,DD8
      TYPE COMPLEX AE,XI,XI1,XI2,FI,FI1,FI2,HI2 ,CC1,CC2,CC3,CC4,CC5
      TYPE COMPLEX A,EIGG,EVAL,EVEC
      TYPE COMPLEX AZ
C
      IF(IORD-1) 10,20,50
   10 WRITE (51,15) ,IORD
   15 FORMAT(/4X,40HIMPOSSIBLE ORDER OF MATRIX ELEMENTS    =,I10 )
      STOP
   20 DO 40 I=1,NN
      DO 30 J=1,NN
      AE(I,J)=A(I,J,2)*AZ+A(I,J,1)
   30 CONTINUE
   40 CONTINUE
      GO TO 80
   50 DO 70 I=1,NN
      DO 60 J=1,NN
      AE(I,J)=AZ*(A(I,J,3)*AZ+A(I,J,2))+A(I,J,1)
   60 CONTINUE
   70 CONTINUE
   80 RETURN
      END
```

```
      SUBROUTINE CKEVEC
C
C         THIS SUBROUTINE DETERMINES THE ACCEPTABILITY OF AN EIGENVECTOR
C
      DIMENSION AE(20,20)
      COMMON A(20,20,3),EIGG(40,2),EVAL(40),EVEC(20,40),NN,IORD,IE,NEIG,
     1IMAX,ISAC,IPRINT,IVEC,EP1,EP2,EP3
      COMMON /GBLOC2/AE,XI,XI1,XI2,FI,FI1,FI2,FX0,HI2,YXI1,YXI2,XS2,II
     1,IJ,IKJ  ,CC1,CC2,CC3,CC4,CC5,DD1,DD2,DD3,DD4,DD5,DD6,DD7,DD8
      TYPE COMPLEX A,EIGG,EVAL,EVEC
      TYPE COMPLEX AE,XI,XI1,XI2,FI,FI1,FI2,HI2 ,CC1,CC2,CC3,CC4,CC5
      DD2=0.0
      DO 100 I=1,NN
      CC2=(0.0,0.0)
      D1=0.0
      DO 80 K=1,NN
C
C         SUM ROW K * EIGENVECTOR
C
      CC1=AE(I,K)*EVEC(K,IJ)
      A1=CABS(CC1)
      CC2=CC2+CC1
C
C         FIND MAXIMUM ELEMENT IN SUM
C
      D1=MAX1F(A1,D1)
   80 CONTINUE
      A1=CABS(CC2)
C
C         DETERMINE IF SUM IS ZERO RELATIVE TO MAXIMUM ELEMENT
C
      IF (A1-EP3*D1) 100,100,150
  100 CONTINUE
      GOTO 200
C
C         SET FLAG STATING EIGENVECTOR UNACCEPATBLE
C
  150 DD2=1.0
  200 RETURN
      END
```

```
      SUBROUTINE FUNCEV(AZ,BZ)
C
C         THIS SUBROUTINE EVALUATES FUNCTION AT PRESENT ITERATE VALUE
C
      DIMENSION AE(20,20)
      COMMON A(20,20,3),EIGG(40,2),EVAL(40),EVEC(20,40),NN,IORD,IE,NEIG,
     1IMAX,ISAC,IPRINT,IVEC,EP1,EP2,EP3
      COMMON /GBLOC2/AE,XI,XI1,XI2,FI,FI1,FI2,FX0,HI2,YXI1,YXI2,XS2,II
     1,IJ,IKJ  ,CC1,CC2,CC3,CC4,CC5,DD1,DD2,DD3,DD4,DD5,DD6,DD7,DD8
      TYPE COMPLEX A,EIGG,EVAL,EVEC
      TYPE COMPLEX AE,XI,XI1,XI2,FI,FI1,FI2,HI2 ,CC1,CC2,CC3,CC4,CC5
      TYPE COMPLEX AZ,BZ,AX
C
C         EVALUATE POLYNOMIAL ELEMENTS AT XI
C
   10 CALL CALFUNC(AZ)
C
C         COMPUTE VALUE OF DETERMINANT
C
   20 CALL CDTERM(NN,AE,BZ,KER,20)
   25 GO TO (60,30),KER
   30 WRITE (51,40) ,IJ,II,AZ
   40 FORMAT(//4X,42HERROR IN FUNCTION EVALUATION OF EIGENVALUE,I5,2X,
     112HAT ITERATION ,I5,2X,3HXI=,C(E22.11,E22.11))
      STOP
   60 IKC=IJ-1
C
C         DETERMINE IF APOSTERIORI SCALING IS NECESSARY
C
      IF(BZ-1.0E+90) 68,65,65
   65 DD8=1.0
   68 CC5=BZ
      IF (IKC) 90,90,70
   70 AX=(1.0,0.0)
C
C         SUPPRESSION OF ROOTS PREVIOUSLY CALCULATED
C
      DO 80 I=1,IKC
      AX=AX*(AZ-EVAL(I)/XS2)
   80 CONTINUE
      BZ=BZ/AX
   90 RETURN
      END
```

```
      SUBROUTINE ROMAT
C
C         THIS SUBROUTINE RETURNS PROBLEM TO ORIGINAL FORM IF
C         APOSTERIORI SCALING HAS OCCURED
C
      DIMENSION AE(20,20)
      COMMON A(20,20,3),EIGG(40,2),EVAL(40),EVEC(20,40),NN,IORD,IE,NEIG,
     1IMAX,ISAC,IPRINT,IVEC,EP1,EP2,EP3
      COMMON /GBLOC2/AE,XI,XI1,XI2,FI,FI1,FI2,FX0,HI2,YXI1,YXI2,XS2,II
     1,IJ,IKJ  ,CC1,CC2,CC3,CC4,CC5,DD1,DD2,DD3,DD4,DD5,DD6,DD7,DD8
      TYPE COMPLEX AE,XI,XI1,XI2,FI,FI1,FI2,HI2 ,CC1,CC2,CC3,CC4,CC5
      TYPE COMPLEX A,EIGG,EVAL,EVEC
      IF(IORD-1.0) 10,10,40
   10 DO 30 I=1,NN
      DO 20 J=1,NN
      A(I,J,1)=A(I,J,1)*XS2
   20 CONTINUE
   30 CONTINUE
      GO TO 100
   40 XS22=XS2*XS2
      DO 60 I=1,NN
      DO 50 J=1,NN
      A(I,J,1)=A(I,J,1)*XS22
      A(I,J,2)=A(I,J,2)*XS2
   50 CONTINUE
   60 CONTINUE
  100 RETURN
      END
```

```
      SUBROUTINE ACCEPT
C
C         THIS SUBROUTINE DETERMINES IF PRESENT ITERATE IS ACCEPTABLE
C
      DIMENSION AE(20,20)
      COMMON A(20,20,3),EIGG(40,2),EVAL(40),EVEC(20,40),NN,IORD,IE,NEIG,
     1IMAX,ISAC,IPRINT,IVEC,EP1,EP2,EP3
      COMMON /GBLOC2/AE,XI,XI1,XI2,FI,FI1,FI2,FX0,HI2,YXI1,YXI2,XS2,II
     1,IJ,IKJ  ,CC1,CC2,CC3,CC4,CC5,DD1,DD2,DD3,DD4,DD5,DD6,DD7,DD8
      TYPE COMPLEX AE,XI,XI1,XI2,FI,FI1,FI2,HI2 ,CC1,CC2,CC3,CC4,CC5
      TYPE COMPLEX A,EIGG,EVAL,EVEC
      IF(II-1) 10,10,20
C
C         INITIALIZE VARIABLES AT FIRST ITERATION
C
   10 YXI1=CABS(XI1)
      YXI2=CABS(XI2)
      IKJ=1
   20 YXI=CABS(XI)
   30 XN=ABSF(YXI-YXI1)
C
C         CHECK FOR RELATIVE CONVERGENCE
C
   40 IF (XN/YXI-EP1) 41,41,120
   41 WRITE (51,45)
   45 FORMAT(5X,34HRELATIVE CONVERGENCE CRITERION MET )
      GO TO 110
C
C         CHECK FOR ABSOLUTE CONVERGENCE
C
  120 IF (CABS(CC5)-EP2) 121,121,90
  121 WRITE (51,122)
  122 FORMAT(5X,34HABSOLUTE CONVERGENCE CRITERION MET   )
C
C         SET FLAG TO STATE PROBLEM HAS CONVERGED
C
  110 IKJ=2
      GO TO 100
C
C         SET VALUES FOR NEXT ITERATION
C
   90 YXI2=YXI1
      YXI1=YXI
  100 RETURN
      END
```

```
      SUBROUTINE SCALE2
C
C         THIS SUBROUTINE DOES THE APOSTERIORI SCALING
C
      DIMENSION AE(20,20)
      COMMON A(20,20,3),EIGG(40,2),EVAL(40),EVEC(20,40),NN,IORD,IE,NEIG,
     1IMAX,ISAC,IPRINT,IVEC,EP1,EP2,EP3
      COMMON /GBLOC2/AE,XI,XI1,XI2,FI,FI1,FI2,FX0,HI2,YXI1,YXI2,XS2,II
     1,IJ,IKJ  ,CC1,CC2,CC3,CC4,CC5,DD1,DD2,DD3,DD4,DD5,DD6,DD7,DD8
      TYPE COMPLEX AE,XI,XI1,XI2,FI,FI1,FI2,HI2 ,CC1,CC2,CC3,CC4,CC5
      TYPE COMPLEX A,EIGG,EVAL,EVEC
      XIR=CABS(XI)
C
C         DETERMINES SCALE FACTOR, CHARACTERISTIC OF LOG(XI), BASE 2
C
   10 XS2=XIR.AND.3777000000000000B
      XS2=XS2.OR.400000000000B
C
C         SET FLAG TO STATE SCALING HAS OCCURED
C
      DD1=1.0
      IF(IORD-1.0) 70,70,100
   70 DO 90 I=1,NN
      DO 80 J=1,NN
C
C         TRANSFORM MATRICES OF COEFFICIENTS IF LINEAR
C
      A(I,J,1)=A(I,J,1)/XS2
   80 CONTINUE
   90 CONTINUE
      GO TO 15
  100 XS22=XS2*XS2
      DO 120 I=1,NN
      DO 110 J=1,NN
C
C         TRANSFORM MATRICES OF COEFFICIENTS IF QUADRATIC
C
      A(I,J,1)=A(I,J,1)/XS22
      A(I,J,2)=A(I,J,2)/XS2
  110 CONTINUE
  120 CONTINUE
```

```
C
C         SETUP LAST THREE VALUES TO START NEW PROBLEM
C
   15 XI=XI/XS2
   20 CALL FUNCEV(XI,FI)
      XI1=XI1/XS2
   30 CALL FUNCEV(XI1,FI1)
      XI2=XI2/XS2
   40 CALL FUNCEV(XI2,FI2)
C
C         CALCULATE NECESSARY VARIABLES TO INITIALIZE NEW PROBLEM
C
      HI2=(FI1-FI2)/(XI1-XI2)
   50 YXI2=CABS(XI2)
      YXI1=CABS(XI1)
      WRITE (51,60) ,XS2
   60 FORMAT(/5X,33HEIGENVALUE  SCALED, SCALE FACTOR=,E15.7)
      RETURN
      END
```

```
      SUBROUTINE SCALE1
C
C         THIS SUBROUTINE DOES THE APRIORI SCALING
C
      DIMENSION AE(20,20)
      DIMENSION C(20)
      COMMON A(20,20,3),EIGG(40,2),EVAL(40),EVEC(20,40),NN,IORD,IE,NEIG,
     1IMAX,ISAC,IPRINT,IVEC,EP1,EP2,EP3
      COMMON /GBLOC2/AE,XI,XI1,XI2,FI,FI1,FI2,FX0,HI2,YXI1,YXI2,XS2,II
     1,IJ,IKJ  ,CC1,CC2,CC3,CC4,CC5,DD1,DD2,DD3,DD4,DD5,DD6,DD7,DD8
      TYPE COMPLEX AE,XI,XI1,XI2,FI,FI1,FI2,HI2 ,CC1,CC2,CC3,CC4,CC5
      TYPE COMPLEX A,EIGG,EVAL,EVEC
C
      X=(1.0,0.0)
      DO 60 J=1,NN
      DO 30 I=1,NN
C
C         SUM POLYNOMIAL COEFFICIENTS FOR ROW J
C
      C(I)=CABS(A(J,I,3)+A(J,I,2)+A(J,I,1))
   30 CONTINUE
      XMAX=C(1)
C
C         FIND MAXIMUM ELEMENT IN ROW J
C
      DO 40 I=2,NN
      XMAX=MAX1F(XMAX,C(I))
   40 CONTINUE
C
C         DETERMINE SCALE FACTOR,CHARACTERISTIC OF LOG(XMAX),BASE 2
C
      XMAX=XMAX.AND. 3777000000000000B
      XMAX=XMAX .OR. 400000000000B
C
C         SCALE ROW J
C
      DO 50 I=1,NN
      A(J,I,3)=A(J,I,3)/XMAX
      A(J,I,2)=A(J,I,2)/XMAX
      A(J,I,1)=A(J,I,1)/XMAX
   50 CONTINUE
   60 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE EVEK
C
C         THIS SUBROUTINE CALCULATES THE ASSOCIATED EIGENVECTOR
C
      DIMENSION AE(20,20)
      DIMENSION B(20,20),C(20,20),X(20)
      COMMON A(20,20,3),EIGG(40,2),EVAL(40),EVEC(20,40),NN,IORD,IE,NEIG,
     1IMAX,ISAC,IPRINT,IVEC,EP1,EP2,EP3
      COMMON /GBLOC2/AE,XI,XI1,XI2,FI,FI1,FI2,FX0,HI2,YXI1,YXI2,XS2,II
     1,IJ,IKJ  ,CC1,CC2,CC3,CC4,CC5,DD1,DD2,DD3,DD4,DD5,DD6,DD7,DD8
      TYPE COMPLEX A,EIGG,EVAL,EVEC
      TYPE COMPLEX AE,XI,XI1,XI2,FI,FI1,FI2,HI2 ,CC1,CC2,CC3,CC4,CC5
      TYPE COMPLEX B,C,X
C
      NX = NN-1
C
C         EVALUATE FUNCTION AT EIGENVALUE IJ
C
      CALL CALFUNC(XI )
      IT=1
      ITL=IT
      IR=0
      GO TO 7
    5 IR=IT
      IT=IT+1
      ITL=IT
    7 ITT=IT-1
      IF(ITT) 10,10,50
C
C         SET FIRST POSITION IN EIGENVECTOR=1
C
   10 EVEC(1,IJ)=(1.0,0.0)
C
C         REMOVE COLUMN 1 FROM MATRIX
C
      DO 30 IB=1,NN
      DO 20 IA=1,NX
      B(IB,IA)=AE(IB,IA+1)
   20 CONTINUE
   30 CONTINUE
```

```
C
C         PUT COLUMN 1 INTO SOLUTION VECTOR POSITION
C
      DO 40 IB=1,NN
      B(IB,NN)=-AE(IB,1)
   40 CONTINUE
      GO TO 200
   50 IF(IT-NN) 100,60,60
C
C         SET LAST POSITION IN EIGENVECTOR=1
C
   60 EVEC(NN,IJ)=(1.0,0.0)
C
C         REMOVE LAST COLUMN FROM MATRIX
C
      DO 80 IB=1,NN
      DO 70 IA=1,NX
      B(IB,IA)=AE(IB,IA)
   70 CONTINUE
   80 CONTINUE
C
C         PUT LAST COLUMN INTO SOLUTION VECTOR POSITION
C
      DO 90 IB=1,NN
      B(IB,NN)=-AE(IB,NN)
   90 CONTINUE
      GO TO 200
C
C         SET POSITION IT OF EIGENVECTOR=1
C
  100 EVEC(IT,IJ)=(1.0,0.0)
C
C         REMOVE COLUMN IT FROM MATRIX
C
      DO 130 IB=1,NN
      DO 120 IA=1,ITT
      B(IB,IA)=AE(IB,IA)
  120 CONTINUE
  130 CONTINUE
      DO 150 IB=1,NN
      DO 140 IA=IT,NX
      B(IB,IA)=AE(IB,IA+1)
  140 CONTINUE
  150 CONTINUE
```

```
C
C         PUT COLUMN IT INTO SOLUTION VECTOR POSITION
C
      DO 160 IB=1,NN
      B(IB,NN)=-AE(IB,IT)
  160 CONTINUE
  200 IRR=IR
      IR=IR+1
      IF (IR-NN) 205,205,202
  202 IR=1
      IRR=0
      ITL=-IT
  205 IF (IRR) 210,210,240
C
C         REMOVE FIRST ROW TO COMPLETE SETUP OF REDUCED MATRIX
C
  210 DO 230 IB=1,NX
      DO 220 IA=1,NN
      C(IB,IA)=B(IB+1,IA)
  220 CONTINUE
  230 CONTINUE
      GO TO 350
  240 IF(IR-NN) 280,250,250
C
C         REMOVE LAST ROW TO COMPLETE SETUP OF REDUCED MATRIX
C
  250 DO 270 IB=1,NX
      DO 260 IA=1,NN
      C(IB,IA)=B(IB,IA)
  260 CONTINUE
  270 CONTINUE
      GO TO 350
C
C         REMOVE ROW IR TO COMPLETE SETUP OF REDUCED MATRIX
C
  280 DO 300 IB=1,IRR
      DO 290 IA=1,NN
      C(IB,IA)=B(IB,IA)
  290 CONTINUE
  300 CONTINUE
```

```
      DO 320 IB=IR,NX
      DO 310 IA=1,NN
      C(IB,IA)=B(IB+1,IA)
  310 CONTINUE
  320 CONTINUE
C
C          SOLVE REDUCED SYSTEM
C
  350 CALL JORCOM(C,NX,X,20,20)
C
C          PLACE SOLUTION OF REDUCED SYSTEM INTO EIGENVECTOR
C
      IF(ITT) 360,360,380
  360 DO 370 IA=2,NN
      EVEC(IA,IJ)=X(IA-1)
  370 CONTINUE
      GO TO 450
  380 IF(IT-NN) 410,390,390
  390 DO 400 IA=1,NX
      EVEC(IA,IJ)=X(IA)
  400 CONTINUE
      GO TO 450
  410 DO 420 IB=1,ITT
      EVEC(IB,IJ)=X(IB)
  420 CONTINUE
      DO 430 IB=IT,NX
      EVEC(IB+1,IJ)=X(IB)
  430 CONTINUE
C
C          CHECK SOLUTION TO DETERMINE ACCEPTABILITY
C
  450 CALL CKEVEC
      IF(DD2) 470,455,470
  455 WRITE (51,460) ,(EVEC(IB,IJ),IB=1,NN)
  460 FORMAT(/4X,23HASSOCIATED EIGENVECTOR  ,/(C(E22.11, E22.11)))
      GO TO 500
  470 IF (IR-ITL-1) 200,5,480
  480 WRITE (51,490) , (EVEC(IB,IJ),IB=1,NN)
  490 FORMAT(/4X,38HUNABLE TO FIND EIGENVECTOR, LAST TRY=  ,/(C(E22.11,E
     122.11)))
  500 RETURN
      END
```

APPENDIX III

A. IDENTIFICATION:

Title: A Method for the Solution of the "Generalized Eigenvalue Problem" with Polynomial Matrix Elements.

CO-OP ID: F4-NPGS-GENEIG (1604 F-63)
Category: Mathematical Subroutine
Programmer: R. D. Brunell
Date: June 10, 1966

B. PURPOSE:

To find those values of the complex parameter, z, which satisfy the system of equations:

$$
\begin{aligned}
&A_{11}(z)x_1 + A_{12}(z)x_2 + \dots + A_{1n}(z)x_n = 0 \\
&A_{21}(z)x_1 + A_{22}(z)x_2 + \dots + A_{2n}(z)x_n = 0 \\
&\quad\vdots \qquad\qquad\quad \vdots \\
&A_{n1}(z)x_1 + A_{n2}(z)x_2 + \dots + A_{nn}(z)x_n = 0,
\end{aligned} \tag{1}
$$

where the a_{ij}'s are complex polynomials of degree 2 in z.

This may be written more concisely as

$$H(z)X = 0, \tag{2}$$

where H(z) is an n x n matrix whose elements are the polynomials in z, and X is a complex column vector of n elements. The elements of the set $\{z_i : H(z_i)X_i = 0\}$ are called the eigenvalues, and the X_i are called the associated eigenvectors.

The matrix H(z), because of the form of its elements, may be written in the following form:

$$H(z) = A_o z^2 + A_1 z + A_2 \quad , \tag{3}$$

where the n x n matrices A_i have constant complex elements.

Equation (3) will be useful later when discussing the input for the subroutine.

C. USAGE:

1. Set up for GENEIG

GENEIG is a subroutine and the user must supply a main program which will set up the input data for GENEIG. All communication to and from this subroutine is done through common storage, and the following DIMENSION, COMMON and TYPE statements must be declared by the user.

```
DIMENSION    A(20,20,3),EIGG(40,2),EVAL(40),EVEC(20,40)

COMMON       A,EIGG,EVAL,EVEC,NN,IORD,IE,
             NEIG,IMAX,ISAC,IPRINT,IVEC,EPI,EP2,EP3

TYPE COMPLEX  A,EIGG,EVAL,EVEC
```

2. Input for GENEIG

The three-dimensional array, A, must contain the elements of the matrix H in the manner described below:

$a^{(o)}_{ij}$ = A(I,J,1), $a^{(1)}_{ij}$ = A(I,J,2) and $a^{(2)}_{ij}$ = A(I,J,3),

where $a^{(k)}_{ij}$ is the ij-th element of A_k.

The two-dimensional array, EIGG, must contain the estimated value for the eigenvalues as follows:

EIGG(I,1) will be considered as the estimate to the value of the i-th eigenvalue, and EIGG(I,2) will be considered as the probable precentage error in that guess. The first three values necessary to start the process for the i-th value are computed by

$$z^{(1)}_i = \text{EIGG(I,1)} - \text{EIGG(I,2)*EIGG(I,1)}$$

(4) $$z^{(2)}_i = \text{EIGG(I,1)}$$

$$z^{(3)}_i = \text{EIGG(I,1)} + \text{EIGG(I,2)*EIGG(I,1)} \quad .$$

The array EVEC will contain the associated eigenvectors. The eigenvector associated with the i-th eigenvalue will be found in EVEC(J,I), where J = 1,2,...,NN.

NN is the order of the coefficient matrices.

IORD is the degree of the polynomial elements.

IE is the number of the first eigenvalue to be found.

NEIG is the number of the last eigenvalue to be found.

IMAX is the maximum number of iterations to be allowed.

ISAC is a flag for initial scaling. If ISAC = 0 no scaling will occur.

IPRINT is a flag for printing at every iteration. If IPRINT = 0 only the final results will be printed.

IVEC is a flag for the eigenvector option. If IVEC = 0 no eigenvectors will be computed.

EPI is the relative error bound on the eigenvalues. If

$|z_i - z_{i-1}| / |z_i| < \text{EPI}$ the problem is considered to have converged.

A value of $10^{-4} \leq \text{EPI} \leq 10^{-8}$ has been found to be satisfactory for most cases.

EP2 is an absolute error bound on the function. If $|H(z_i)| < \text{EP2}$ the problem is considered to have converged. A value of $\text{EP2} \approx 10^{-15}$ appears to be satisfactory for those cases for which this test becomes effective.

EP3 is a relative error bound on the eigenvector. If B is the maximum element of $H(z_i)$, then if $|H(z_i)X_i| < |\text{EP3*B}|$ the eigenvector is considered acceptable. A value of $\text{EP3} \approx 10^{-4}$ is satisfactory.

3. Calling Sequence:

Once the common storage has been set up in the user's program it is only necessary to insert the instruction CALL GENEIG. The subroutine will then compute all eigenvalues and eigenvectors, that have been indicated by the input, before returning control to the calling program.

4. Space Required: 4217_{10} excluding common storage

5. Common Storage Required: 4251_{10}

6. Print-Outs: If IPRINT = 0 then at each iteration the number of the iteration, the present value of z_i and $H(z_i)$ will be printed.

   When an eigenvalue is accepted, a message to this effect will be printed along with the accepted value and the function value.

   If IVEC = 0 and an eigenvector is found it will be printed. However, if no acceptable eigenvector is found, a statement to this effect will be printed along with the value of the final attempt.

7. Error Stops: If IORD is found to be other than an integer 1 or 2 the program will print the value of IORD and stop (Exit to Monitor). If the determinant evaluator is unable to complete its operation the code will indicate that this has happened and stop (Exit to Monitor).

8. Input and Output Formats: Not applicable.

9. Input and Output Tape Mountings: Output is to Logical unit 51 (Standard on CO-OP Monitor).

10. Selective Jump and Stop Settings: Not applicable.

11. Timing Examples:

Case	1	2	3	4	5	6
Size of Matrix	3x3	4x4	5x5	8x8	12x12	16x16
No. of Eigenvalues	6	4	5	8	12	4
No. of Eigenvectors	6	4	0	0	0	0
Average No. of Iterations	4.0	8.25	6.0	10.5	18.7	17.5
Total Time (seconds)	3.85	3.85	4.38	16.0	104.	55.28

12. Accuracy:

   Results from this code have been compared with results taken from several papers on the standard eigenvalue problem and they have been as accurate or more accurate on all tests. Also, several problems were run for which exact results were known. On problems with well-defined roots nine or ten figure accuracy was observed. When pathological roots were encountered the accuracy of this code compared very favorably with other methods tested.

13. Cautions to User: The DIMENSION, COMMON and TYPE statements as shown in Section C must be used exactly as indicated.

14. Equipment Configuration: Standard CO-OP Computer with FORTRAN 63 Compiler and CO-OP Monitor System.

15. References:

(a) R. D. Brunell, An Iterative Solution to the Generalized Eigenvalue-Eigenvector Problem, Technical Report/Research Paper No. 69, U. S. Naval Postgraduate School, Monterey, California, 1966.

(b) J. F. Traub, Iterative Methods for the Solution of Equations, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1964.

(c) I. Tarnove, "Determination of Eigenvalues of Matrices Having Polynomial Elements," Journal of SIAM, June 1958.

D. METHOD:

Only a general discussion of the method will be presented here. For a more detailed description see Reference (a).

The heart of the method used in this subroutine is the iterative root-finding technique, which was developed by J. F. Traub (b). In essence the technique is to perform a quadratic fit using the last three iterants and their functional values. The new estimate to the root is defined as that root of the fitting quadratic which is closer to the last iterant. The process converges with almost arbitrary estimates; however, through the use of good estimates a considerable saving in computer time can be realized. Fewer iterations will be necessary and only those eigenvalues of interest need be computed.

In order to avoid convergence to a previously-calculated root a method of suppressing zeros, first suggested by G. E. Forsythe (Reference (c) page 164), was incorporated into the code. This feature can also be used to eliminate re-calculation of roots found on a previous run, especially when they may lie pathologically close to a desired root. This can be accomplished by loading the m known roots into the first m positions in the EVAL array and setting IE equal to m + 1.

To alleviate the problem of floating-point exponential overflow a two-part, *a priori* and *a posteriori*, automatic scaling procedure was devised for this code.

The *a priori* scaling is optional and is controlled by the flag ISAC. This scaling occurs only once and is done prior to any calculations for eigenvalues. The coefficients of each polynomial element are summed and a maximum ,B, is found in each row. The scaling is carried out by dividing each coefficient of every polynomial in a row by 2^{J+1}, where $2^{j} \leq B \leq 2^{J+1}$.

The *a posteriori* scaling is not optional and will occur only when an overflow is sensed. At this point the problem is transformed so that the scaled eigenvalue will be exactly the original eigenvalue divided by 2^{k+1}, where $2^{k} \leq z_i \leq 2^{k+1}$ and z_i is the value of the iterant at the time of the overflow.

The computation of the associated eigenvectors is an optional feature in this code. The basic assumption in this section is that the eigenvalue reduces the rank of the original matrix by 1. Under this assumption 2NN attempts will be made to find an acceptable eigenvector. If all attempts fail, a printout to this effect will be made.

DOCUMENT CONTROL DATA - R&D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY *(Corporate author)* U. S. Naval Postgraduate School Monterey, California	2a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED 2b. GROUP

3. REPORT TITLE

AN ITERATIVE SOLUTION TO THE GENERALIZED EIGENVALUE-EIGENVECTOR PROBLEM

4. DESCRIPTIVE NOTES *(Type of report and inclusive dates)*

5. AUTHOR(S) *(Last name, first name, initial)*

Brunell, Ronald D.

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
12 July 1966	52	

8a. CONTRACT OR GRANT NO.	9a. ORIGINATOR'S REPORT NUMBER(S)
b. PROJECT NO.	TR/RP No. 69
c.	9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)*
d.	

10. AVAILABILITY/LIMITATION NOTICES

Unlimited

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY

13. ABSTRACT

The abstract is as given on the title page.

Security Classification

KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Generalized						
Eigenvalue						
Eigenvector						

INSTRUCTIONS

. ORIGINATING ACTIVITY: Enter the name and address f the contractor, subcontractor, grantee, Department of De-ense activity or other organization (*corporate author*) issuing he report.

a. REPORT SECURITY CLASSIFICATION: Enter the over-ll security classification of the report. Indicate whether Restricted Data" is included. Marking is to be in accord-nce with appropriate security regulations.

b. GROUP: Automatic downgrading is specified in DoD Di-ective 5200.10 and Armed Forces Industrial Manual. Enter he group number. Also, when applicable, show that optional arkings have been used for Group 3 and Group 4 as author-zed.

. REPORT TITLE: Enter the complete report title in all apital letters. Titles in all cases should be unclassified. f a meaningful title cannot be selected without classifica-ion, show title classification in all capitals in parenthesis mmediately following the title.

. DESCRIPTIVE NOTES: If appropriate, enter the type of eport, e.g., interim, progress, summary, annual, or final. Give the inclusive dates when a specific reporting period is overed.

. AUTHOR(S): Enter the name(s) of author(s) as shown on r in the report. Enter last name, first name, middle initial. military, show rank and branch of service. The name of he principal author is an absolute minimum requirement.

. REPORT DATE: Enter the date of the report as day, onth, year; or month, year. If more than one date appears n the report, use date of publication.

a. TOTAL NUMBER OF PAGES: The total page count hould follow normal pagination procedures, i.e., enter the umber of pages containing information.

b. NUMBER OF REFERENCES: Enter the total number of eferences cited in the report.

a. CONTRACT OR GRANT NUMBER: If appropriate, enter he applicable number of the contract or grant under which he report was written.

b, 8c, & 8*d*. PROJECT NUMBER: Enter the appropriate ilitary department identification, such as project number, ubproject number, system numbers, task number, etc.

a. ORIGINATOR'S REPORT NUMBER(S): Enter the offi-ial report number by which the document will be identified nd controlled by the originating activity. This number must e unique to this report.

b. OTHER REPORT NUMBER(S): If the report has been ssigned any other report numbers (*either by the originator r by the sponsor*), also enter this number(s).

0. AVAILABILITY/LIMITATION NOTICES: Enter any lim-itations on further dissemination of the report, other than those imposed by security classification, using standard statements such as:

(1) "Qualified requesters may obtain copies of this report from DDC."

(2) "Foreign announcement and dissemination of this report by DDC is not authorized."

(3) "U. S. Government agencies may obtain copies of this report directly from DDC. Other qualified DDC users shall request through

____ ."

(4) "U. S. military agencies may obtain copies of this report directly from DDC. Other qualified users shall request through

____ ."

(5) "All distribution of this report is controlled. Qual-ified DDC users shall request through

____ ."

If the report has been furnished to the Office of Technical Services, Department of Commerce, for sale to the public, indi-cate this fact and enter the price, if known.

11. SUPPLEMENTARY NOTES: Use for additional explana-tory notes.

12. SPONSORING MILITARY ACTIVITY: Enter the name of the departmental project office or laboratory sponsoring (*pay-ing for*) the research and development. Include address.

13. ABSTRACT: Enter an abstract giving a brief and factual summary of the document indicative of the report, even though it may also appear elsewhere in the body of the technical re-port. If additional space is required, a continuation sheet shall be attached.

It is highly desirable that the abstract of classified reports be unclassified. Each paragraph of the abstract shall end with an indication of the military security classification of the in-formation in the paragraph, represented as *(TS), (S), (C), or (U)*.

There is no limitation on the length of the abstract. How-ever, the suggested length is from 150 to 225 words.

14. KEY WORDS: Key words are technically meaningful terms or short phrases that characterize a report and may be used as index entries for cataloging the report. Key words must be selected so that no security classification is required. Identi-fiers, such as equipment model designation, trade name, military project code name, geographic location, may be used as key words but will be followed by an indication of technical con-text. The assignment of links, roles, and weights is optional.

D FORM 1 JAN 64 1473 (BACK)

UNCLASSIFIED

Security Classification

DISTRIBUTION LIST

Documents Department
General Library
University of California
Berkeley, California 94720

Lockheed-California Company
Centeral Library
Dept. 77-14, Bldg. 170, Plt. B-1
Burbank, California 91503

Naval Ordnance Test Station
China Lake, California
Attn: Technical Library

Serials Dept., Library
University of California, San Diego
La Jolla, California 92038

Aircraft Division
Douglas Aircraft Company, Inc.
3855 Lakewood Boulevard
Long Beach, California 90801
Attn: Technical Library

Librarian
Government Publications Room
University of California
Los Angeles, California 90024

Librarian
Numerical Analysis Research
University of California
405 Hilgard Avenue
Los Angeles, California 90024

Chief Scientist
Office of Naval Research
Branch Office
1030 East Green Street
Pasadena, California 91101

Commanding Officer and Director
U. S. Navy Electronics Lab. (Library)
San Diego, California 92152

General Dynamics/Convair
P.O. Box 1950
San Diego, California 92112
Attn: Engineering Library
Mail Zone 6-157

Ryan Aeronautical Company
Attn: Technical Information
Services
Lindbergh Field
San Diego, California 92112

General Electric Company
Technical Information Center
P.O. Drawer QQ
Santa Barbara, California 93102

Library
Boulder Laboratories
National Bureau of Standards
Boulder, Colorado 80302

Government Documents Division
University of Colorado Libraries
Boulder, Colorado 80304

The Library
United Aircraft Corporation
400 Main Street
East Hartford, Connecticut 06108

Documents Division
Yale University Library
New Haven, Connecticut 06520

Librarian
Bureau of Naval Weapons
Washington, D. C. 20360

George Washington University Library
2023 G Street, N. W.
Washington, D. C. 20006

National Bureau of Standards Library
Room 301, Northwest Building
Washington, D. C. 20234

Director
Naval Research Laboratory
Washington, D. C. 20390
Attn: Code 2027

University of Chicago Library
Serial Records Department
Chicago, Illinois 60637

Documents Department
Northwestern University Library
Evanston, Illinois 60201

The Technological Institute, Library
Northwestern University
Evanston, Illinois 60201

Librarian
Purdue University
Lafayette, Indiana 47907

Johns Hopkins University Library
Baltimore
Maryland 21218

Martin Company
Science-Technology Library
Mail 398
Baltimore, Maryland 21203

Scientific and Technical Information Facility
Attn: NASA Representative
P.O. Box 5700
Bethesda, Maryland 20014

Documents Office
University of Maryland Library
College Park, Maryland 20742

The Johns Hopkins University
Applied Physics Laboratory
Silver Spring, Maryland
Attn: Document Librarian

Librarian
Technical Library, Code 245L
Building 39/3
Boston Naval Shipyard
Boston, Massachusetts 02129

Massachusetts Institute of Technology
Serials and Documents
Hayden Library
Cambridge, Massachusetts 02139

Technical Report Collection
303A, Pierce Hall
Harvard University
Cambridge, Massachusetts 02138
Attn: Mr. John A. Harrison, Librarian

Alumni Memorial Library
Lowell Technological Institute
Lowell, Massachusetts

Librarian
University of Michigan
Ann Arbor, Michigan 48104

Gifts and Exchange Division
Walter Library
University of Minnesota
Minneapolis, Minnesota 55455

Reference Department
John M. Olin Library
Washington University
6600 Millbrook Boulevard
St. Louis, Missouri 63130

Librarian
Forrestal Research Center
Princeton University
Princeton, New Jersey 08540

U. S. Naval Air Turbine Test Station
Attn: Foundational Research Coordinator
Trenton, New Jersey 08607

Engineering Library
Plant 25
Grumman Aircraft Engineering Corp.
Bethpage, L. I., New York 11714

Librarian
Fordham University
Bronx, New York 10458

U. S. Naval Applied Science Laboratory
Technical Library
Building 291, Code 9832
Naval Base
Brooklyn, New York 11251

Librarian
Cornell Aeronautical Laboratory
4455 Genesee Street
Buffalo, New York 14225

Central Serial Record Dept.
Cornell University Library
Ithaca, New York 14850

Columbia University Libraries
Documents Acquisitions
535 W. 114 Street
New York, New York 10027

Engineering Societies Library
345 East 47th Street
New York, New York 10017

Library-Serials Department
Rensselaer Polytechnic Institute
Troy, New York 12181

Librarian
Documents Division
Duke University
Durham, North Carolina 27706

Ohio State University Libraries
Serial Division
1858 Neil Avenue
Columbus, Ohio 43210

Commander
Philadelphia Naval Shipyard
Philadelphia, Pennsylvania 19112
Attn: Librarian, Code 249c

Steam Engineering Library
Westinghouse Electric Corporation
Lester Branch Postoffice
Philadelphia, Pennsylvania 19113

Hunt Library
Carnegie Institute of Technology
Pittsburgh, Pennsylvania 15213

Documents Division
Brown University Library
Providence, Rhode Island 02912

Central Research Library
Oak Ridge National Laboratory
Post Office Box X
Oak Ridge, Tennessee 37831

Documents Division
The Library
Texas A & M University
College Station, Texas 77843

Librarian
LTV Vought Aeronautics Division
P.O. Box 5907
Dallas, Texas 75222

Gifts and Exchange Section
Periodicals Department
University of Utah Libraries
Salt Lake City, Utah 84112

Defense Documentation Center (DDC)
Cameron Station
Alexandria, Virginia 22314
Attn: IRS (20 copies)

FOREIGN COUNTRIES

Engineering Library
Hawker Siddeley Engineering
Box 6001
Toronto International Airport
Ontario, Canada
Attn: Mrs. M. Newns, Librarian

Exchange Section
National Lending Library for
Science and Technology
Boston Spa
Yorkshire, England

The Librarian
Patent Office Library
25 Southampton Buildings
Chancery Lane
London W. C. 2., England

Librarian
National Inst. of Oceanography
Wormley, Godalming
Surrey, England

Dr. H. Tigerschiöld, Director
Library
Chalmers University of Technology
Gibraltargatan 5
Gothenburg S, Sweden

International Computation Centre
Palazzo degli Uffici-Zona dell' E.U.R.
Rome, Italy